Tasks

Choice 1.

I have created programming exercises. You have to create a github repository and upload your code there. I'll evaluate the functionality, your coding style, accordance to the coding standards and code coverage by unit tests.

Exercise 1:

Finish the string calculator code.

```
1. Create a simple String calculator with a method signature:
```

int Add(string numbers)

```
2. The method can take up to two numbers, separated by commas, and will
return their sum.
```

for example “” or “1” or “1,2” as inputs. (for an empty string it will return 0)

Hints:

- Start with the simplest test case of an empty string and move to one and two numbers
- Remember to solve things as simply as possible so that you force yourself to write tests you did not think about
- Remember to refactor after each passing test

```
3. Allow the Add method to handle an unknown amount of numbers
4. Allow the Add method to handle new lines between numbers (instead of
commas).
```

the following input is ok: “1\n2,3” (will equal 6) the following input is NOT ok: “1,\n” (not need to prove it - just clarifying)

```
5. Support different delimiters
* to change a delimiter, the beginning of the string will contain a separate
line that looks like this: “//[delimiter]\n[numbers…]” for example
“//;\n1;2” should return three where the default delimiter is ‘;’ .
* the first line is optional. all existing scenarios should still be
supported
* Calling Add with a negative number will throw an exception “negatives not
allowed” - and the negative that was passed.
* if there are multiple negatives, show all of them in the exception
message.
```

STOP HERE if you are a beginner. Continue if you can finish the steps so far in less than 30 minutes.

```
6. Numbers bigger than 1000 should be ignored, so adding 2 + 1001 = 2
7. Delimiters can be of any length with the following format:
“//[delimiter]\n” for example: “//[***]\n1***2***3” should return 6
```

————————————————————————————————

```
8. Allow multiple delimiters like this:
```

“*[delim1][delim2]\n” for example “*[*][%]\n1*2%3” should return 6.

```
9. make sure you can also handle multiple delimiters with length longer than
one char
```

Exercise 2:

- Add Logging Abilities to your new String Calculator (to an ILogger.Write()) interface (**you will need a mock**).
- Every time you call Add(), the sum result will be logged to the logger.
- When calling Add() and the logger throws an exception, the string calculator should notify an IWebservice of some kind that logging has failed with the message from the logger’s exception (you will need a mock and a stub).

Exercise 3:

Create a Password verifications class called “PasswordVerifier”.

Add the following verifications to a master function called “Verify()”

- password should be larger than 8 chars
- password should not be null
- password should have one uppercase letter at least
- password should have one lowercase letter at least
- password should have one number at least

Each one of these should throw an exception with a different message of your choosing

- Add feature: Password is OK if at least three of the previous conditions is true
- Add feature: password is never OK if item “should have one lowercase letter at least” is not true.
- Assume Each verification takes 1 second to complete. How would you solve that tests can run faster?

Exercise 4:

A measuring machine measures certain parameters of a workpiece and creates a pdf report file. That pdf file will be sent to customers. Customer want to be sure that the pdf file is valid in other words it

was generated by the software at the specified time and no one manipulated the file. So that pdf file has to be digitally signed.

For the exercises use a coding standard like https://google.github.io/styleguide/javaguide.html or http://tinyurl.com/pmz4xc5 You have to create unit tests.

Choice 2.

Write an essay. **The essay must be unpublished, original work**. I offer 4 topics - you need to select only one

1. Service Level Agreement in software maintenance
2. Quality assurance for machine learning
3. Visual testing - hint https://applitools.com/blog/visual-testing
4. Smart contracts in automotive industry

Formal requirements at: https://icessd.uni-miskolc.hu/content/7/7_2.docx. **The essay must be more than 5 pages long**

Deadlines:

Please decide whether you want to write an essay or implement the coding excercises. Put your choice to the shared document https://docs.google.com/spreadsheets/d/1xkIua-5uW5hOQ_B88uXlhsD6i9OL2KOa3nPV4WleU9k/edit?usp=sharing

1. All the exercises and essays must be finished by **19:00 6th May 2020**.
2. Deadline can be extended by your request, but final deadline is 19:00 13th May 2020.

External Links

Eclipse tutorial

Understanding Test Driven Development

Unit Testing Best Practices

Practical Refactoring

Understanding Mock Objects

Unit Testing Worst Practices

Kent Beck on creating JUnit

Lecture on Design Patterns

Scrum training

Unit testing with Junit

Unit testing with Junit

Git and Github tutorial

Textbooks

Java Coding Standard by Oracle or download the local copy of the document

Daniel Galin: Software Quality Assurance From theory to implementation

The Art of Unit Testing

Sample excercises can be downloaded.

Sample code

First unit testing lesson

Handouts

Power point slides

Suggested schedule

Week ending 27 April 2020

Watch the video series called JUnit 5 Basics

Create your own JUnit tests and be confident how to use it. If you have any problem my code is available, see the sample code section

Week ending 3 April 2020

Watch the video Understanding Mock Objects

Create a sample code using Mock objects. You can use this link.

Week ending 10 April 2020

Watch this tutorial about git and guthub.

Create a github repository and your mock sample code.

Week ending 17 April 2020

More videos about git https://www.youtube.com/playlist?list=PLpL2ONl1hMLtlY1Y7YJNcA5zumvaITLYs

https://youtu.be/LPT7v69guVY

Week ending 24 April 2020

Please watch first 6 videos from
https://www.youtube.com/watch?v=vhYK3pDUijk&list=PLGLfVvz_LVvSuz6NuHAzpM52qKM6bPlCV

Week ending 1 May 2020

Please watch second 6 videos from
https://www.youtube.com/watch?v=vhYK3pDUijk&list=PLGLfVvz_LVvSuz6NuHAzpM52qKM6bPlCV

Week ending 8 May 2020

Please watch last 6 videos from
https://www.youtube.com/watch?v=vhYK3pDUijk&list=PLGLfVvz_LVvSuz6NuHAzpM52qKM6bPlCV

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:quality_assurance_for_information_technology

Last update: **2020/05/08 05:41**